Compositional CBR via Collaborative Filtering

Stefano Aguzzoli* and **Paolo Avesani** and **Paolo Massa**
ITC-irst - 38050 Povo, Italy
{aguzzoli,avesani,massa}@itc.it

Abstract

We introduce an application combining CBR and collaborative filtering techniques in the music domain. We describe a scenario in which the classical collaborative filtering recommendation algorithm suffers from serious drawbacks: this scenario stresses the difference between a single-interaction case and a dynamically growing user profile. We set up a framework meant to extend collaborative filtering for compositional recommendation systems where cases does not explicitly yield the amount of overlapping items needed by classical filtering.

Introduction

The notion of intelligent sales support (Cunningham *et al.* 2000; Schafer *et al.* 1999) is going to be widely recognized as a new challenge for e-commerce applications. The new scenario promoted by the Internet has increased the opportunities to customize products; now, it is possible to deliver goods or services close as much as possible to a user's needs.

The pre-sales phase in the context of electronic commerce is being revisited, because the role of the customer has changed. In the past, a customer could only select a product from a set catalogue made in advance by the supplier organization. Now, the customer is directly involved in the process of assembling the product, a task that requires the combination of a set of predefined components in accordance with user preferences.

The music market (Hayes & Cunningham 2000) represent one of the most notable scenarios in which this evolution is occurring. In the past, the delivered goods were compact discs containing compilations (that is, selected collections) of audio tracks; nowadays, it is possible to buy individual tracks rather than entire precompiled CDs. In spite of this change, the notion of "compilation" is still relevant, because e-commerce systems enable the purchase of single tracks, but the delivery still relies on compact discs as media. In this scenario, the role of editor of compilations has moved from the market expert to the final user.

New Web portals devoted to the sale of sound tracks[1] allow each user to create his own compilation. Usually, the assembling process is based on an iterative step that adds a sound track selected from the result set of a query to a repository of music.

From this perspective, the advantage introduced by the personalization of the product could bring some drawbacks. Firstly, the simple iterative step of adding one track after another quickly becomes boring, as an average compilation includes some 15 tracks. Secondly, query engines on track repositories are not able to provide querying over many important features of music, because tracks tagging is time consuming and user dependent. Finally, there is no way to formulate a query to a repository in which each individual track is enriched with information on what is an appropriate context for its inclusion in a compilation.

Giving the users the opportunity to assemble their own compilation, enables the portal to collect important knowledge about the sound tracks. Indeed, every single compilation may be considered as an instance of the *genre of use* related to its sound tracks. This kind of genres are not the usual categories used to classify music (rock, jazz, classical, and so on); the genres of use represents all the different perspectives along with a user can look at music. Following this approach, one can imagine to have the standard compilations of "rock" rather than "jazz music", but at the same time one could have compilations of "driving west coast" music rather than "a night with my girl" music, which may possibly put together tracks that would be kept separated under a traditional classification system.

From this perspective, a repository of compilations may be considered a powerful case base to support both the editing of personalized compilations and the detection of the genres of use introduced by customers.

In the following, we examine how case-based reasoning can be applied to support product personaliza-

*Current address: Computer Science Department, University of Milan; email: stefano@gongolo.usr.dsi.unimi.it.

[1] See as examples the following Web sites: www.mixfactory.com, www.musicalacarte.net.

tion (Stahl & Bergmann 2000; Schmitt & Bergmann 1999; Wilke, Lenz, & Wess 1998) by taking advantage of collaborative filtering techniques (Goldberg *et al.* 1992; Resnick & Varian 1997). After presenting the motivations for this work, we briefly introduce CoCoA, a recommendation system for music compilations. The following section gives a detailed explanation of the design of the case-based retrieval engine used by CoCoA, which is based on the collaborative filtering approach. We then show the results of an empirical evaluation carried out on a dataset derived from EachMovie (McJones 1997). The paper concludes with a brief discussion of the results, including a comparison with the Singular Value Decomposition technique (Golub & Loan 1983; Forsythe, Malcolm, & Moler 1977).

Motivations

As described above, the customer is involved in the process of assembling their own compilation, by adding one track after another. Some of the Web portals that distribute music started to enhance this interaction by providing a recommendation utility. By selecting the recommendation option, the customer can access a list of tracks that the system has detected as relevant for the current partial compilation. Even though this kind of systems relies on repositories of compilations, there is no way for the customer to browse them.

In the recommendation approach, the suggestion (the sound track) is provided without the context (the compilation) that could help the user to receive a sort of explanation of why the track proposed by the system has been selected. Furthermore, the customer cannot directly interpret and reuse what other customers have done in the past by looking at the existing compilations and deriving a new one from them.

The first motivation of our work was to introduce the CBR loop in the context of compositional e-commerce, where a customer is required to collect a set of components to fulfill the pre-sales phase. The following section illustrates two interaction models based on case-based reasoning designed for CoCoA.

A further motivation underlying our work is to integrate into the CBR loop a retrieval engine so that does not require to address the content description in order to compute the similarity measure. In the domain of music this aspect is crucial for two reasons: firstly, because the definition of music categories does not meet a general agreement; and secondly, because the features that usually describe a sound track do not discriminate them between the different genres of use.

The integration of CBR and Collaborative Filtering in the context of compositional e-commerce presents a couple of drawbacks. The first is related to the choice of the notion of case: we cannot associate a case description to a single user, because the experience is represented by a single compilation. Each compilation can be considered as an episode where the goal of collecting a set of tracks following a given criteria has been achieved. Looking at one compilation as a single case is one of our working assumptions. At the same time, from a collaborative filtering perspective, we consider the single compilation as the profile of a virtual user. The second drawback is related to the notion of rating. Usually the collaborative filtering techniques rely on the correlation between two user profiles where the ratings allow the system assess their similarity. In the case of compilations, an explicit rating of the sound tracks is not available and we can simply assume that all the tracks belonging to the same compilation have an equal rating.

Summarizing, we have to cope with the following two drawbacks:

1. The case is a single compilation, its dimension is fixed and very small compared to the set of all tracks. Compare this with the profile of the movies graded by a user. The information in this profile grows indefinitely as the user keeps on grading newly seen movies. In the compilation scenario, the tracks are given once and for all, and two compilations are not likely to have any significant overlap of common tracks.
2. The case does not contain any graded rating of the tracks occurring therein: simply, a track is there or not. We do not know from the case if a track is actually very significant for the case or mildly significant or even it is just to be regarded as rumor. Moreover, all we know of a track is its name.

Figure 1: **CoCoA GUI:** the CoCoA graphical user interface.

Given the assumptions illustrated above, we have to deal with the problem of making the collaborative filtering approach effective in a situation where drawbacks 1 and 2 are present. The objective of this work is to extend the standard collaborative filtering approach to enable its use in a CBR loop in the context of compositional e-commerce.

Before introducing our approach to this challenge, we briefly illustrate the CoCoA system, which represents

the first practical application of our work.

The CoCoA System

CoCoA, a COmpilation COmpiler Advisor, has been designed as a Web service that supports the editing of a compilation by means of a repository of sound tracks and a case base of compilations.

The system architecture allows the sound track providers to plug this compositional facility into their existing sales software. The management and distribution of the sound tracks is not part of CoCoA, which supports only the assembling process and the management of the case base repository, i.e. the database of virtual compilations, containing only track indexes.

A running version of this Web service has been deployed for the Karadar[2] Web site, a provider of free classical music. Starting from the catalogue of Karadar (a set of around 2000 sound tracks) CoCoA allows the user to collect a set of tracks in a compilation, while the download of the final product is left again to the Karadar Web site.

The bootstrap of the case base has been performed by looking at the log files of the Web site. The mining of this kind of data allowed for the detection of the sequences of downloads concerning 5 to 12 sound tracks. For each of these sequences, a compilation was stored in the case base.

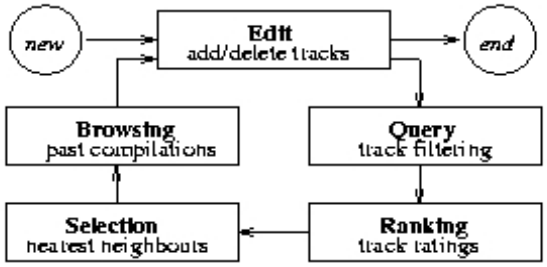

Figure 2: **User Adaptation:** the CBR interaction supports the retrieval of past similar compilations, the choice concerning the reuse of a such kind of information remains in charge of the user.

From a CBR perspective, the system implements two models of interaction: the recommend model and the complete model. The recommend model is depicted in figure 2, which summarizes the main steps of the interaction. The user looking at the repository of sound tracks can iteratively delete or add new tracks to his own compilation. On demand, the user can decide to invoke the recommend utility in order to add new tracks taken from past compilations. First, the current partial compilation is addressed to the retrieval engine as an implicit query where the terms are the sound tracks currently included in the collection. A second step performs a ranking process to sort the compilations of the case base with respect to the given query. A third step selects the closest compilations, which are presented to the user. A fourth step, under the control of the user, allows the browsing of the compilations returned by the query; the selection of tracks for the completion of the user's collection.

[2]See the URL `http://www.karadar.com`.

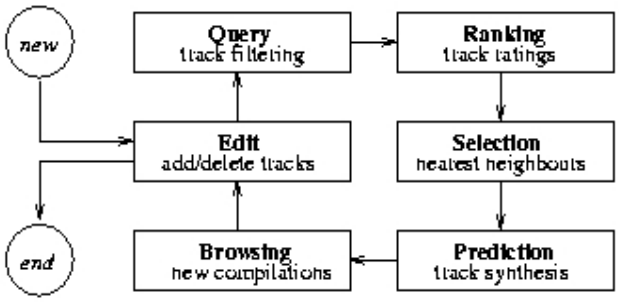

Figure 3: **System Completion:** the CBR steps of retrieval and reuse are managed automatically by the system; the revision and the refinement of proposed solution are still in charge of the end user.

A second interaction model, namely the completion model, is essentially organized as the previous one. The main difference concerns a supplementary step between the selection and the browsing steps. A prediction process enable the system to perform a synthesis of new compilations providing a possible completion of the partial collection assembled by the user. In this case, the user will browse a set of "virtual" compilations, since the case base does not necessarily include any of them. Of course, the completion of the final collection still remains the decision of the user.

After having presented the interaction models, we now explain how a recommendation engine supporting this kind of model has been designed, and how the drawbacks highlighted in the previous section have been overtaken.

The recommendation engine

The recommendation engine represents the core of a CBR system for compositional e-commerce. It takes as inputs the partial collection being assembled and the repository of past compilations. Its output is a subset of the case base and a ranking of the selected compilations.

To illustrate in detail how, given these inputs, we obtain a sound output, let us go through a step-by-step definition of the basic elements of the computational method. For the sake of simplicity, we continue to refer to the music domain, without loss of generality.

Definition 1 (Sound Tracks) $T = \{t_i\}$ *is a finite collection of components, namely sound tracks.*

It is worthwhile to notice that the definition of sound tracks does not require to provide a feature value specification. Any individual sound track is represented only by its id code.

Definition 2 (Track Compilation) *Let n be some fixed integer much smaller than the cardinality of T. Then $C = \{c_i = \{t_{i1}, \ldots, t_{in}\} \mid t_{ij} \in T\}$ is a finite collection of sound tracks.*

Although a compilation entails the notion of sequence, in the following we will not take this information into account. From now on, we assume that the compositional problem is concerned only with the notion of collection.

Definition 3 (Active Compilation) *$A = \{a_i = \{t_{i1}, \ldots, t_{im}\} \mid t_{ij} \in T,\ m \leq n\}$ is a finite collection of sound tracks at most of the same cardinality of a track compilation.*

The active compilation plays the role of the recommendation query when a collection of track compilations is organized in a case base. Before introducing how a single compilation can be viewed as a user profile from a collaborative filtering perspective, let us see what kind of information can be extracted from a case base.

Throughout this paper by CB we shall denote the subset of C forming the current case base.

Definition 4 (Frequency of track pairs) *Let $s : 2^T \rightarrow 2^C$ be a function that given a subset S of sound tracks returns the subset of compilations that include them: $s(S) = \{c \in CB | S \subseteq c\}$. Then $f_\nu : 2^T \rightarrow [0,1] \subset \mathcal{R}^+$ allows to obtain the frequency of a subset of sound tracks: $f_\nu(S) = |s(S)||CB|^{-1}$. An approximation of the sound track probability can be derived from the track frequency:*

$$\hat{P}(t_i) = f_{t_i}(t_i) = f_\nu(\{t_i\}) \quad f_{t_i}(t_j) = \frac{|s(\{t_i, t_j\})|}{|CB|} = f_{t_j}(t_i)$$

Using this formulation we can derive also the approximation of the conditional probability:

$$\hat{P}(t_i|t_j) = \frac{f_{t_j}(t_i)}{f_{t_j}(t_j)} = \frac{f_{t_i}(t_j)}{f_{t_j}(t_j)}$$

The above definition introduced the functions $f_{t_i}(t)$ that can be considered a kind of track profile where $f_{t_i}(t_j)$ says how much the track t_i "likes" the track t_j, i.e. the rating of t_j formulated by t_i.

Now, we need to summarize a kind of compilation profile starting from the $f_{t_i}(t)$ that belongs to a given compilation.

Definition 5 (Compilation Synthetic Function) *Given a function $\mu : 2^T \rightarrow [0,1]^T$ and a compilation $c \in CB \subset C$, where $c = \{t_1, \ldots, t_n\}$, a characteristic function can be defined as:*

$$f_c : T \rightarrow [0,1], \quad f_c(t) = \mu(t_1, \ldots, t_n)(t)$$

The function $f_c(t)$ allows to summarize a compilation profile that does not restrict its definition to the subset of track components. $f_c(t)$ is defined over all T; it does mean that we can interpret the value $f_c(\bar{t})$, where $\bar{t} \notin c$, as an estimate of how much the compilation c likes the track $\bar{t}$. In this way we can cope with the scarce overlap of compilations (see drawback 1). Further, the range of any f_c is the interval $[0,1]$ thus providing the needed graduation of the ratings (see drawback 2).

In the following, we propose an hypothesis for μ although we are aware that its choice is related to a definition of a loss function. More details on the evaluation criteria will be illustrated in the section dedicated to the experiments.

Definition 6 (μ Function) *Let μ be a linear composition:*

$$\mu(d_1, \ldots, d_n)(t) = \sum_{d_j} \kappa(d_j) \frac{f_{d_j}(t)}{f_{d_j}(d_j)}$$

where $\kappa(d_j)$ is a weight to balance the contribution of the single components of the given compilation.

Although the objective of the parameter $\kappa(d_j)$ is to measure how much a track is representative for the genre of use of the compilation which it belongs to, for the sake of simplicity in the following we assume $\kappa(d_j)$ to be a constant.

Once a synthetic function to model the compilation is available, we can simply rely on a linear process to assess the similarity and to rank the case base.

Definition 7 (Selection Process) *A function $f_r : A \times 2^C \rightarrow 2^C$ to select the nearest neighbours compilations, given a distance measure d and a threshold k:*

$$f_r(a_i, CB) = \{c_i | d(a_i, c_i) \leq k,\ c_i \in CB,\ CB \subset C\}$$

To compare two compilations as usually performed in collaborative filtering, we can refer to the Pearson correlation coefficient. This measure can now be effective, because the drawbacks 1 and 2 described earlier have been removed. The compilation profiles are widely defined over T increasing their overlap. Moreover the compilation profiles, even the implicit and tacit rating derived from the compilation, can differentiate the agreement on different sound tracks through the synthesis of frequency pairs.

Definition 8 (Pearson Correlation Coefficient) *Let $a \in A$ be an active compilation and $c \in CB$ a past compilation, given a range $D \subseteq T$ a function $d : A \times CB \rightarrow [-1,1]$ returns the linear correlation factor between two compilations:*

$$d(a,c) = \frac{\sum_{t \in D}(\mu_a(t) - \bar{\mu}_a)(\mu_c(t) - \bar{\mu}_c)}{\sqrt{\sum_{t \in D}(\mu_a(t) - \bar{\mu}_a)^2 \sum_{t \in D}(\mu_c(t) - \bar{\mu}_c)^2}}$$

where $\bar{\mu}_a$ and $\bar{\mu}_c$ respectively denote the mean value of μ_a and μ_c over D.

Associated with the Pearson correlation coefficient an algorithm has been developed, called *agave-t* , implementing the schema illustrated above. The major enhancement introduced by this algorithm is that two compilations can be compared even if they do not share

any track. Moreover, the comparison can occur also between sound tracks that do not belong to any of the two compilations, which are the majority.

We developed a different version of the algorithm, *agave-c* , restricting the range of comparison to the subset of tracks belonging to the active compilation or the track compilation. This restriction reduces the problem of dimensionality suffered by the approach based on user profiles. For the same reason, we have designed a third version of the algorithm, *agave-cvs* , that implements a different comparison based on vector similarity.

Definition 9 (Vector Similarity Coefficient) *Let $a \in A$ be an active compilation and $c \in CB$ a past compilation, given a range $D = c \cup (a \cap T)$ a function $d : A \times CB \rightarrow \mathcal{R}^+$ return the cosine of the angle between two compilations:*

$$d(a,c) = \sum_{t \in D} \frac{\mu_a(t)}{\sqrt{\sum_{t \in D} \mu_a(t)^2}} \frac{\mu_c(t)}{\sqrt{\sum_{t \in D} \mu_c(t)^2}}$$

The variation introduced with the *agave-c* algorithm has a significant impact on the computational complexity. The restriction on the dimensionality of D allows us to reduce the complexity to a linear dependency on the size of the case base, because in $O(|D||CB|)$ the term $|D|$ becomes a constant factor.

Related Works

Recently, many works have been published in the area of Collaborative Filtering, but the focus of our paper is slightly different. While in standard recommendation systems the goal is to predict with accuracy the affinity of the user to a specific good, in the context of compositional e-commerce this goal is extended to a collection of goods. For this reason a direct comparison of the proposed approach with the main recommendation system is not possible.

We prefer to move the analysis of the related work to a basic technique underlying many applications in the area of information retrieval and collaborative filtering: the singular value decomposition (SVD) (Berry & Dumais 1995; Sarwar *et al.* 2000; Billsus & Pazzani 1998; Pryor 1998). Although other methods have been derived from SVD, such as the latent semantic analysis (LSI) (Deerwester *et al.* 1990) and the related probabilistic variation (PLSI) (Hofmann 2001), we limit the comparison to SVD because it is representative of the approaches based on dimensionality reduction.

SVD can be viewed as a technique for deriving a set of uncorrelated indexing variables or factors (and these can be our *genres of use*); each track and compilation is represented by its vector of factor values. Note that by virtue of the dimension reduction, it is possible for compilations with different tracks to be mapped into the same vector of factor values.

A byproduct of the SVD approach is the possibility to estimate a correlation factor between two collections that do not share any of their components. This feature is strongly related to the challenge engaged with the compositional e-commerce.

Let us summarize how the SVD works looking at our chosen domain of music. The starting point of SVD is a very sparse matrix X of $c \times t$ (compilations $\times$ tracks) where every cell contains 1 if that track is contained in that compilation and 0 otherwise. SVD allows to decompose this rectangular matrix into three other matrices of a very special form (the resulting matrices contain "singular vectors" and "singular values").

These special matrices yield a breakdown of the original relationships into linearly independent components or factors. In general, many of these components are very small, and may be ignored, leading to an approximate model with many fewer dimensions. In this reduced model all the track-track, compilation-compilation and track-compilation similarity is now approximated by values in this smaller number of dimensions. The result can be still represented geometrically by a spatial configuration in which the dot product or cosine between vectors representing two objects corresponds to their estimated similarity.

Evaluation results

After the presentation of our approach, based on the mix of CBR and CF, and the state of the art concerning the basic technique to implement a recommendation system, we focus our attention to the empirical analysis.

In the following, we illustrate how we selected the dataset to perform a comparative analysis, then we explain how we have designed the experiments and the evaluation setup. Finally, a discussion of the results tries to summarize the main contribution of this work.

Dataset setup

To evaluate our approach we needed information about compilations of objects (music tracks, movies, etc. . .). The CoCoA system is going to be deployed on the Web; in the meantime, we cannot exploit the case base that will be available as soon as it achieves a significant number of users. Waiting for a real case base of compilations, we had to look for an alternative solution.

The community that is working on the recommendation system provides some dataset based on the notions of user profiles and goods. But the concept of *compilation* is slightly different from *profile* because the former is an aggregation of items usually with a small and limited size, while the latter is allowed to grow indefinitely. A user profile is effective as long as it increases, and very often the recommendation accuracy is related to its size.

Moreover, a further difference distinguishes a user profile from a compilation: all the items of a collection are strongly related by a sort of homogeneous factor. We can assume that, even if not explicit, it does exist a common feature that takes the same value for all the items of the collection. We are actually assuming that every compilation has a rationale for its existence, even

if this rationale is not explicitly stated or even known. On the contrary, a user profile does not provide such a kind of information concerning the relationship among a selection of items rated by a single user.

For these reasons, we could not use a traditional Collaborative Filtering dataset, so we have designed a specific one for our purpose. The synthesis of the dataset has been derived starting from Eachmovie (McJones 1997), a dataset composed by 2811983 ratings expressed by 72916 users on 1648 movies. A related dataset, the IMDB, supports an extended content description for every movie like the year of release and the indication of the genres.

We generated a dataset of movie collections giving a genre to each of them. A genre has been defined following a criteria of uniformity along one or more features for every movie of the collection. We defined 15 different genres: before 90', 90'-93', 94', 95', 96', Action, Animation, Art_Foreign, Classic, Comedy, Drama, Family, Horror, Romance, Thriller. This choice allows that the same movie belongs to at least a couple of collection genres.

To simulate the way in which people select the movies to aggregate them in collections, we performed a pseudo-random extraction using the data of Eachmovie. We derived the distribution from the frequency of the rated movies. Also, the number of compilations of each genre is proportional to the number of ratings people gave to movies belonging to that genre.

The total number of compilations is 2995 made out of 1086 movies and each compilation contains 10 movies.

Setup experiment

The task selected for the experiment is the retrieval of the 10 nearest neighbours among the train set compilations for every test set compilation. We have created 10 crossfolds dividing randomly the 2995 compilations in a train set of 2395 compilations and a test set of 600.

To evaluate the interaction model illustrated above, we needed to test the usual way of invoking the recommendation utility: the typical scenario sees the user insert some movies (or items, in the general case) and then ask for similar past compilations, hopefully of the genre sketched in the active compilation. For this reason it is crucial to properly manage short queries intended as partial compilations.

To test this aspect, we have designed different experiments on the same dataset, in which we changed the shadowed portion of every test compilation. More precisely, we replicated the standard test step many times taking into account respectively only 3, 6 and 9 movies of each compilation in the test set. In this way we can simulate the evolution of the active compilation and the different complexity of the related query. The challenge is to achieve good accuracy performance overall when only a sketch of the compilation is available, i.e. the system is dealing with "short" queries.

The performance evaluation measures the accuracy in terms of percentage of recommended compilations with the same genre of the active one. For every test compilations we have extracted the 1nn. The computation of the mean error over the 10nn has given similar results.

Discussion of results

In figure 4 you can see our algorithm *agave-t* compared to *SVD* with different number of kept dimensions. Although the parameter for the dimensionality reduction has been optimized, namely $k = 200$, *agave-t* still outperforms SVD. This result provides an evidence that the proposed approach is an effective technique to reduce the sparseness coefficient that passes from 99% to 25%.

Anyway it is worth noticing that the improvement introduced with *agave-t* is not balanced by the on line computational effort. The theoretical online complexity, crucial for time performance, gives opposite results: $O(k|CB|)$ for *SVD* and $O(|T||CB|)$ for *agave-t* . This comparison is quite crucial because in general k is much smaller than $|T|$.

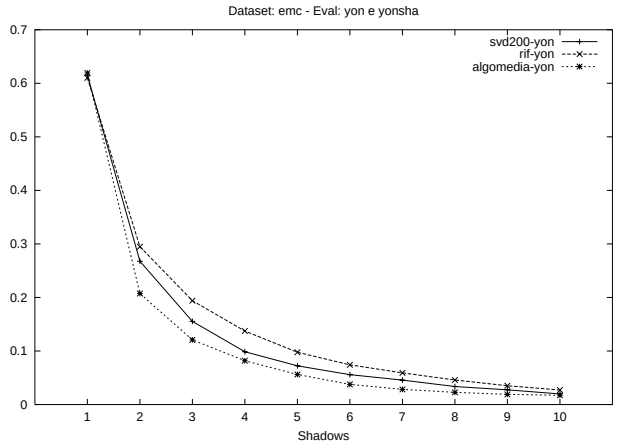

Figure 4: Mean error of *genre of use* matching between test compilation and the 1nn train compilation recommended. The experiments are relatives to different shadow sizes.

However, assuming it is viable to slightly relax the accuracy constraint to give much more attention to the time response, the proposed approach can achieve interesting results. Taking this perspective we also have tried a version of the algorithm, called *agave-c* , that reduces the computation of the Pearson correlation coefficient between the test and train compilations considering only the movies of those compilations instead of all the movies. In this way, the mean complexity is reduced to $O(2\overline{c}|CB|)$ where $\overline{c}$ is the average number of movies in a compilation (10, in our experiments). As expected, the accuracy error increases but, differently from the previous case, the enhancement achieved on the time response is much more significant than the loss of accuracy. Moreover the accuracy of SVD decreases

much faster than *agave-c* . To highlight this aspect we have compared the achieved results with *svd15* that has a comparable online complexity. The results concerning *agave-c* are depicted in figure 5.

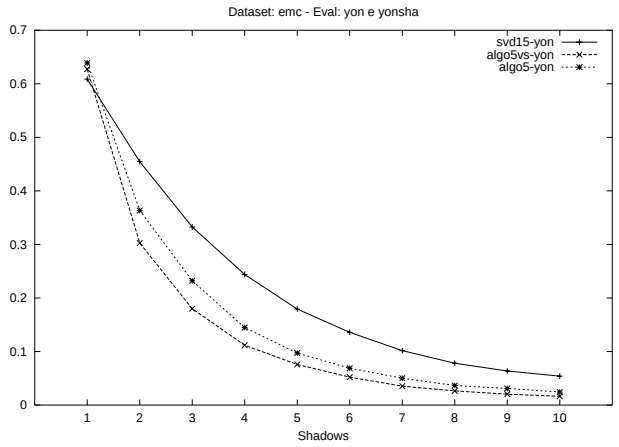

Figure 5: Mean error of *genre of use* matching between test compilation and the 1nn train compilation recommended. The experiments are relatives to different shadow size. The algorithms compared are *agave-c* *agave-cvs* and *SVD* with different k. The loss in accuracy of *agave-c* compared to *agave-t* is counterbalanced by a meaningful improvement of the time response.

The plot shows also *agave-cvs* , a version of *agave-c* in which, to compute the similarity between two compilations, we substitute Pearson correlation coefficient with the cosine of the angle of the vectors representing the two compilations.

This simpler measure (used for instance by SVD) works even better. The reason is related to the dimensionality reduction: the reduced extent prevents the problem of the curse of dimensionality. In this case the Pearson correlation coefficient is less effective because the notion of sparseness does not apply: let us remind the reader that on average we have to compare two vectors of 10-15 components.

As previously stated, our main goal is to reach a certain level of accuracy especially when trying to find a compilation to recommend in reply to a query containing few movies. The results show that, even in presence of partial query compilations, the accuracy of the recommended compilation is good. This is significant especially when compared to Collaborative Filtering algorithms that work well only after the acquisition of a meaningful history interaction to achieve a rich user profile.

From the point of view of CBR this result allows to exploit the collaborative filtering technique without the restriction of the user profile but enabling the association between case and a single user interaction. In this way the case base becomes the repository of the interaction episodes and the user is free to play different roles, i.e. to assemble compilations not necessarily related to each other.

	svd200	*agave-t*	*agave-c*	*agave-cvs*	*svd15*
3	0.152	0.124	0.238	0.186	0.352
6	0.058	0.038	0.072	0.050	0.147
9	0.026	0.018	0.027	0.017	0.069

Table 1: The results depicted in figure 4 and figure 5

Future directions

First of all, we hope as soon as possible to make available a real case base of compilations through the deployment of CoCoA[3].

This step is the premise for the refinement of the evaluation setup that has to overcome the constraint of fixed genres. We need to deal with a situation where every single compilation might be representative of a new genre of use. The final goal is to have the opportunity to assess the genre proximity.

The hypothesis for $\mu_c(t)$ illustrated in this paper represents a preliminary solution to the aggregation problem of the different track profiles encoded by $f_i(t_j)$. A deeper understanding concerning alternative schemas has to be achieved. For example: how to balance the contribution of the different track profiles is still an open problem. We are already taking into account graph-based clustering techniques to detect how much a single track can be considered representative of a genre of use.

Finally we have not yet designed the completion function to support the related interaction model. The delivery of such a kind of support could create the premises for an interesting comparison from a CBR point of view between the recommender and the completion models.

Acknowledgment

We would like to thank Anna Perini, Michele Nori, Gianluca Mameli and Marco Grimaldi for their important role in the design and the development of CoCoA.

The *Each-Movie* database were generously provided by the System Research Center of Digital Equipment Corporation that makes it available for research.

References

Berry, M., and Dumais, S. 1995. Using linear algebra for intelligent information retrieval. *SIAM Review 37(4)* 573–595.

Billsus, D., and Pazzani, M. J. 1998. Learning collaborative information filters. Technical report, AAAI.

[3]A beta version is available at the following address `http://cocoa.itc.it/`

Cunningham, P.; Bergmann, R.; Schmitt, S.; Traphoener, R.; and Breen, S. 2000. Websell: Intelligent sales assistants for the world wide web. Technical report, Trinity College Dublin.

Deerwester, S.; Dumais, S.; Furnas, G.; Landauer, T.; and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 391–407.

Forsythe, G. E.; Malcolm, M. A.; and Moler, C. B. 1977. *Computer Methods for Mathematical Computations*. Englewood Cliffs, NJ 07632, USA: Prentice-Hall.

Goldberg, D.; Nichols, D.; Oki, B.; and Terry, D. 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 35(12):61–70.

Golub, G. H., and Loan, C. F. V. 1983. *Matrix Computations*. Baltimore, MD, USA and Oxford, England: The Johns Hopkins University Press and North Oxford Academic.

Hayes, C., and Cunningham, P. 2000. Smart radio: Building music radio on the fly. In *Expert Systems 2000, Cambridge, UK*.

Hofmann, T. 2001. Probabilistic latent semantic analysis. 177–196.

McJones, P. 1997. Eachmovie collaborative filtering data set, dec systems research center. http://research.compaq.com/SRC/eachmovie/.

Pryor, M. 1998. The effects of singular value decomposition on collaborative filtering. Senior Honors Thesis PCS-TR98-338, Dartmouth College.

Resnick, P., and Varian, H. 1997. Recommender systems. *Communications of the ACM* 40(3):56–58.

Sarwar, B. M.; Karypis, G.; Konstan, J. A.; and Riedl, J. 2000. Application of dimensionality reduction in recommender system – a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*.

Schafer, J.; Konstan, J.; J.; and Riedl. 1999. Recommender systems in e-commerce. In *Proceeding of the ACM Conference on Electronic Commerce*.

Schmitt, S., and Bergmann, R. 1999. Applying case-based reasoning technology for product selection and customization in electronic commerce environments. In *12th Bled Electronic Commerce Conference*.

Stahl, A., and Bergmann, R. 2000. Applying recursive CBR for the custumization of structured products in an electronic shop. In *EWCBR*, 297–308.

Wilke, W.; Lenz, M.; and Wess, S. 1998. Intelligent sales support with CBR. In *Case-Based Reasoning Technology*, 91–114.